Experiment No.: 02

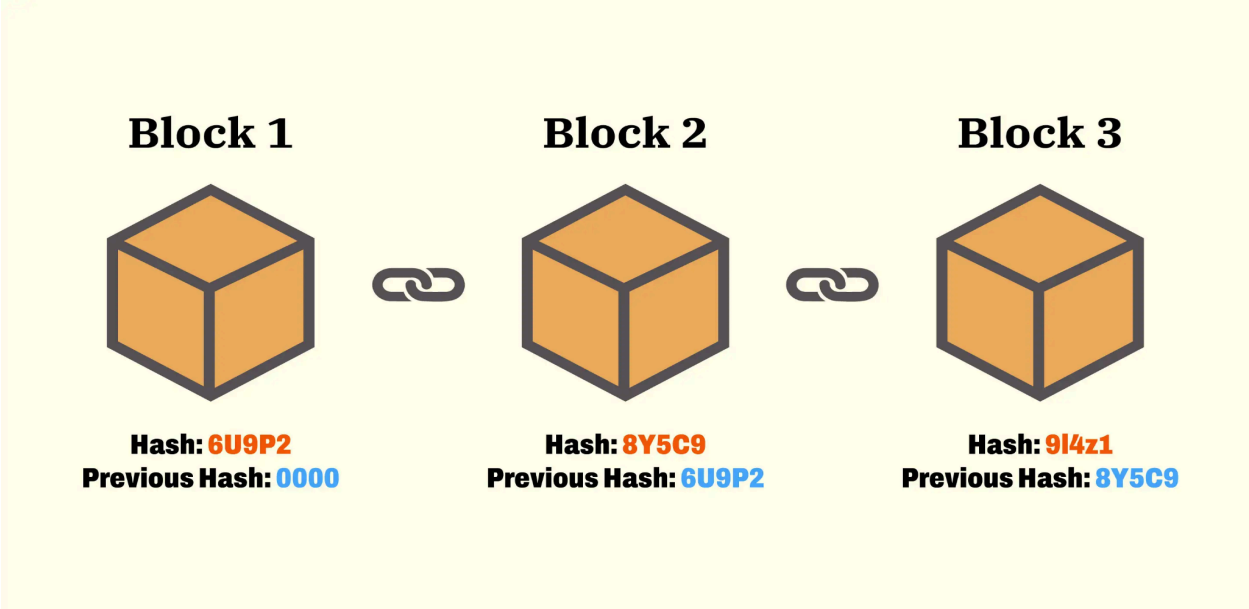
Aim: Create a Blockchain using Python

Task to be performed :

1. Make a copy of this **Google Colab Notebook**
2. Try to solve the errors in given Program
3. After successful execution of the Program in Colab Notebook.
   - Add a method, create_Transactions
   - Mine the block only when the transaction list is not null.
   - Remove the transactions from the list of transactions before mining.
   - Modify the method, proof_of_work() to search for the golden nonce
   - Cryptographic Puzzle is to have **"000" leading zeros** in the Block Hash
4. Download the code - **blockchain.py**
5. Update the code to incorporate the changes in step 3 to the code in step 4.
6. Follow the steps in **Manual** to demonstrate the working of Blockchain using Flask and Postman.

Theory:

What is Blockchain ?



Blockchain is a shared, immutable digital ledger, enabling the recording of transactions and the tracking of assets within a business network and providing a single source of truth.

Blockchain operates as a decentralized distributed database, with data stored across multiple computers, making it resistant to tampering. Transactions are validated through a consensus mechanism, ensuring agreement across the network.

In blockchain technology, each transaction is grouped into blocks, which are then linked together, forming a secure and transparent chain. This structure guarantees data integrity and provides a tamper-proof record, making blockchain ideal for applications like cryptocurrencies and supply chain management.

The key benefit of blockchain lies in its ability to provide security, transparency and trust without relying on traditional intermediaries, such as banks or other third parties. Its design reduces the risk of fraud and errors, making it especially valuable in industries where secure transactions are critical, including finance and healthcare. In addition, blockchain helps businesses improve efficiency and reduce costs by streamlining processes and enhancing accountability.

All nodes or systems are connected peer to peer (P2P) in a network. This distribution is on a conceptual network level.

Key features of blockchain technology

1] Distributed ledger technology

All network participants have access to the distributed ledger and its immutable record of transactions. This shared ledger records transactions only once, eliminating the duplication of effort typical of traditional business networks.

2] Immutable records

No participant can change or tamper with a transaction after it's been recorded in the shared ledger. If a transaction record includes an error, a new transaction must be added to reverse the error, and both transactions are then visible.

3] Smart contracts

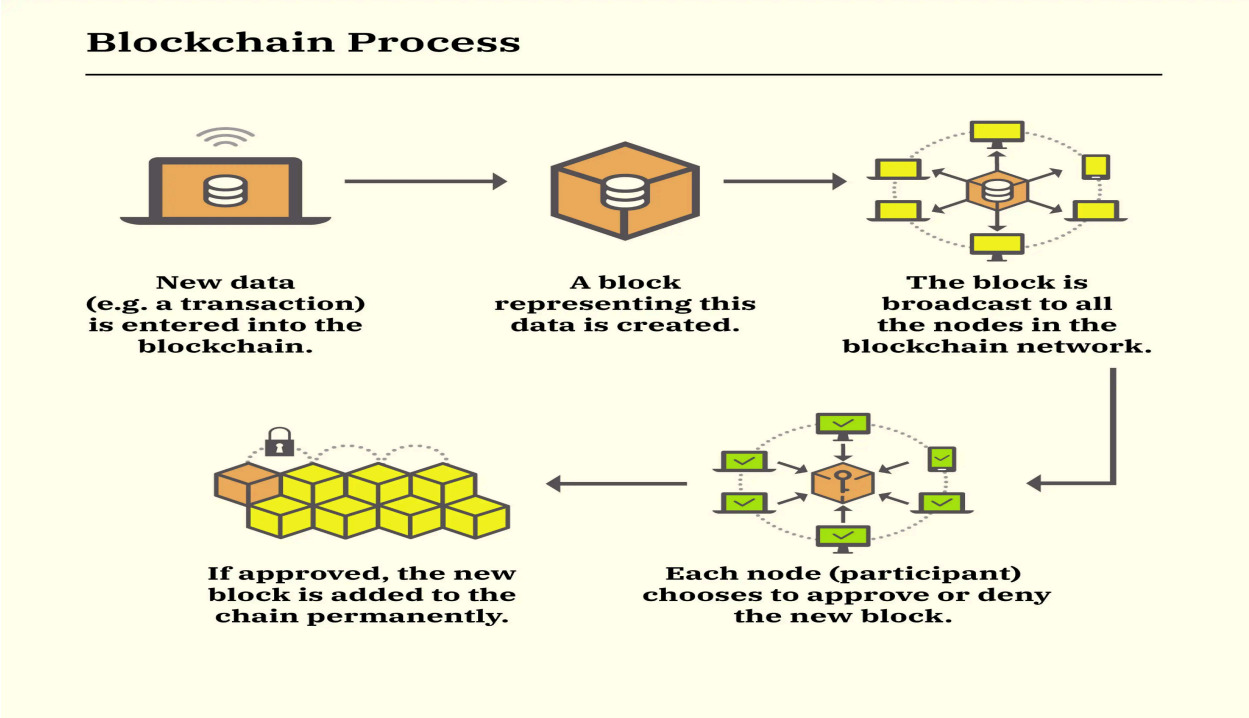
Smart contracts are self-executing agreements stored on the blockchain, where the terms are written in code and automatically executed when predefined conditions are met. They can be used for various purposes, such as transferring corporate bonds or triggering travel insurance

payouts. By automating these processes, smart contracts speed up transactions, reduce the need for intermediaries and ensure transparency and security.

4] Public key cryptography

Public key cryptography is a method used to secure transactions and data on the blockchain by leveraging two cryptographic keys: a public key and a private key. The public key serves as an address for receiving cryptocurrency or data, while the private key is a confidential key that grants control over the associated digital assets. The private key holder can authorize transactions, providing security and verifying ownership, while the public key allows others to send funds or data to the correct address.

How Blockchain Works



1] Records transactions as blocks

Each transaction is recorded as a “block” of data on the blockchain. These blocks capture key details about the movement of assets, whether tangible (such as a product) or intangible (such as intellectual property). The data within each block includes critical information, such as who, what, when, where, the transaction amount, and specific conditions like the temperature of a food shipment.In addition, each block contains a timestamp, which records the exact moment the transaction is added to the blockchain. This timestamp ensures the chronological order of

transactions and adds an additional layer of verifiability to the data, preventing any retrospective alterations to the recorded information.

2] Connects blocks together

Each block is linked to the previous block and the one after it, creating a secure chain of data. This chain is done through cryptographic hashes, unique identifiers for each block. The hash of a block includes data from the previous block, ensuring the exact sequence and timing of each transaction. The cryptographic hash makes it nearly impossible to alter any block without changing all subsequent blocks, ensuring the integrity of the entire process.

3] Builds an irreversible blockchain

The blocks are grouped in an irreversible chain known as a blockchain. Each new block reinforces the security and validation of the previous one, strengthening the entire chain. This Bitcoin-based architecture is what makes decentralized systems so secure and reliable.

Nodes in the blockchain network validate and maintain the blockchain by confirming each transaction's validity through consensus algorithms, ensuring the system remains secure and immutable. Proof of work (PoW) and proof of stake (PoS) are some of the most commonly used consensus algorithms in blockchain networks, each helping to secure the system while validating transactions.

4] Ensures trust and immutability

With each new block, the blockchain becomes more secure, making it nearly impossible to change past transactions. This immutability provides a trusted, transparent ledger that all network members can rely on, preventing fraud and ensuring that all transaction records are accurate and unchangeable.

Types of blockchain networks

Public blockchain networks: A public blockchain is open for anyone to join and participate in, such as the Bitcoin blockchain. While it offers decentralization, it also comes with drawbacks, including high computational power requirements, lack of transaction privacy and potentially weaker security. These considerations are crucial, especially for enterprise blockchain use cases.

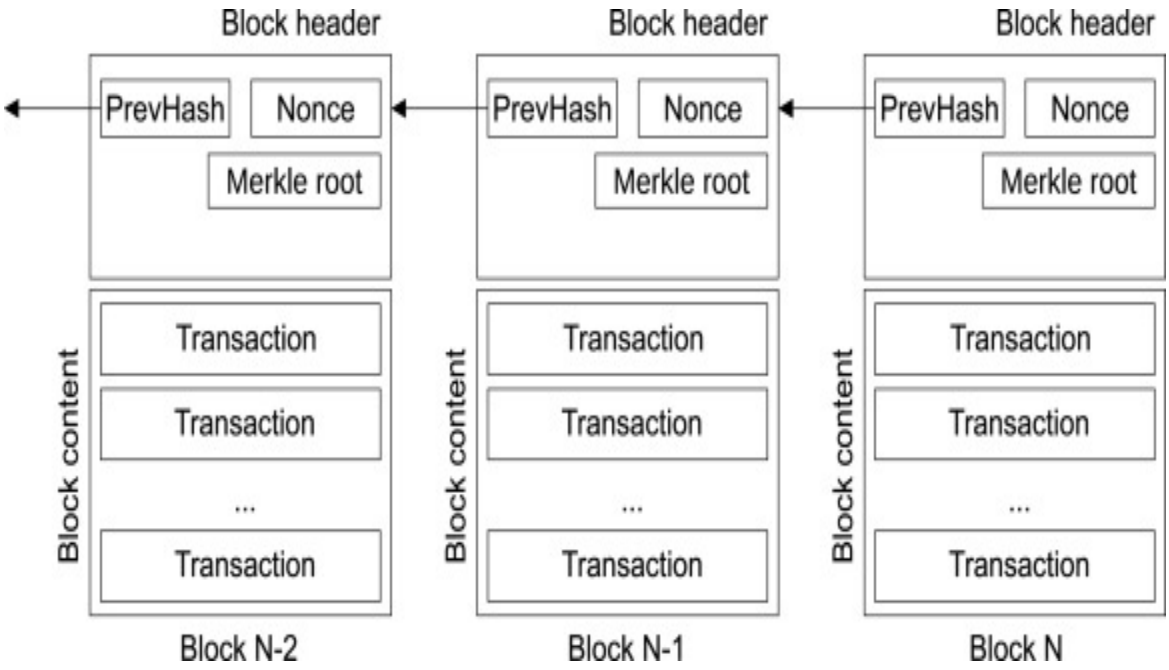
Private blockchain networks: A private blockchain network, similar to a public blockchain network, is a decentralized peer-to-peer network. However, one organization governs the network, controlling who is allowed to participate, run a consensus protocol and maintain the shared ledger. Depending on the use case, this infrastructure can significantly boost trust and confidence between participants. A private blockchain can be run behind a corporate firewall and even be hosted on premises.

Permissioned blockchain networks: Businesses that set up a private blockchain set up a permissioned blockchain network. It is important to mention that public blockchain networks can also be permissioned. This places restrictions on who is allowed to participate in the network and in what record transactions. Participants need to obtain an invitation or permission to join.

Consortium blockchain networks: A group of preselected organizations actively manages a consortium blockchain network and shares the responsibility of maintaining the blockchain. These organizations determine who can submit transactions and access data. This type of network is ideal when multiple parties must collaborate with shared responsibilities. In the energy sector, energy producers and consumers might share data about power usage and distribution.

Block in Blockchain

A block is the fundamental unit of a blockchain that contains a set of validated transactions and cryptographic links to previous blocks, forming an immutable record in the blockchain network.
Each block in the blockchain is a digital container that permanently stores transaction data for the network. When new transactions occur, they are processed and bundled into a block. Once the network validates these transactions, the block is sealed and linked cryptographically to previous blocks. This creates a chain where each block's contents can't be altered without affecting the others.



Block in a Blockchain–

- Blockchain is a linear chain of blocks.
- Each block contains a set of transactions and other essential details.
- Blocks are linearly connected and cryptographically secured.
- Each block header contains the previous block hash, current block hash, nonce, Merkle root, and other details.
- All blocks are connected linearly by carrying the hash of the previous block.
- The previous block hash is used to compute the current block hash.
- The first block with no previous block hash is called “Genesis Block.”
- For adding a new block to the network, the blockchain follows consensus mechanisms like proof of work (PoW), proof of stake (PoS), etc.

Following are the significant elements of a block –

Block Height –

It’s the sequence number of the block in the chain of blocks. Block Height: 1 is the genesis block (first block in the network).

Block Size –

It’s a 4-bytes or 32-bit field that contains the size of the block. It adds size in Bytes. Ex – Block Size: 216 Bytes.

Block Reward –

This field contains the amount rewarded to the miner for adding a block of transactions.

Tx Count –

The transaction counter shows the number of transactions contained by the block. The field has a maximum size of 9 bytes.

Block Header –

The Block header is an 80-Byte field that contains the metadata – the data about the block.

6 components of the Block Header.

- **Time –** It's the digitally recorded moment of time when the block has been mined. It is used to validate the transactions.
- **Version –** It's a 4-bytes field representing the version number of the protocol used. Usually, for bitcoin, it's '0x1'.
- **Previous Block Hash –** It's a 32-bytes field that contains a 256-bits hash (created by SHA-256 cryptographic hashing) of the previous block. This helps to create a linear chain of blocks.
- **Bits –** It's a 4-bytes field that tells the complexity to add the block. It's also known as "difficulty bits." According to PoW, the block hash should be less than the difficulty level.
- **Nonce –** It's a 4-bytes field that contains a 32-bit number. These are the only changeable element in a block of transactions. In PoW, miners alter nonce until they find the right block hash.
- **Merkle Root –** A 32-bytes field containing a 256-bit root hash. It's constructed hierarchically combining hashes of the individual transactions in a block.

Transactions

It's a variable-size field that includes the list of all transactions contained in the block.
Each bitcoin block contains about 2000 transactions. The size of each block is approx 1MB. The size and number of transactions in a block vary in blockchains. It's decided based on network congestion and communication overhead.

Process of Mining

Blockchain mining is a critical component of maintaining and securing blockchain networks. At its core, mining involves validators, often called miners, using their computational power to solve complex mathematical puzzles. The primary purpose of this process is to verify transactions and add them to the blockchain's ledger, a public record of all transactions within the network. As a reward for their efforts, miners receive cryptocurrency tokens, an incentive that ensures the ongoing integrity and functionality of the blockchain. This decentralised validation process distinguishes blockchain from traditional centralised databases, providing higher security and transparency.

Bitcoin mining is an essential process that enables the functioning of the Bitcoin network. It involves miners using powerful computers to solve complex mathematical puzzles. The first miner to solve the puzzle gets the opportunity to add a new block of transactions to the blockchain, which rewards them with a fixed number of bitcoins. This process is known as "proof of work," as it requires significant computational effort to prove that the miner has completed the required work to add the new block.

How the Bitcoin Mining Process Works?

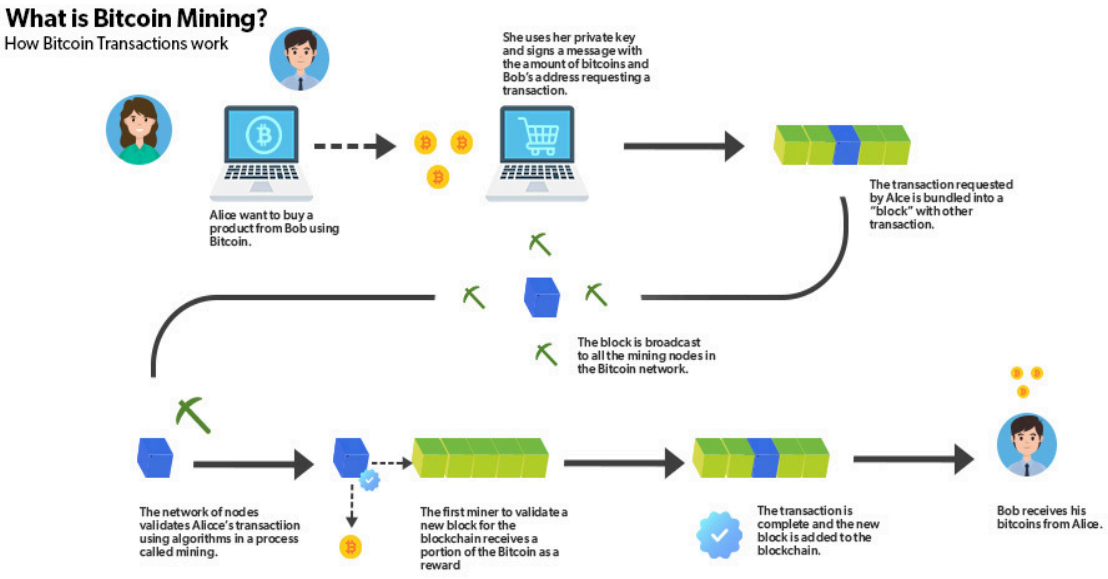
Bitcoin mining is designed to be decentralised and secure.

- **Transaction Collection:** You gather unconfirmed transactions from the Bitcoin network and compile them into a block.
- **Block Header Creation:** You create a block header that includes essential information like the previous block's hash, a Merkle root of the transactions, and a timestamp.
- **Hash Puzzle Solving:** You use trial and error to find a nonce value that results in a hash below the network's difficulty target. The difficulty adjusts every 2016 blocks to maintain a block time of approximately 10 minutes.
- **Block Validation:** Once you solve the puzzle, you broadcast the block to the network. Other nodes verify the block and its transactions before adding it to the blockchain.
- **Reward Distribution:** The successful miner receives a block reward in newly minted Bitcoins and transaction fees for the block.

The nodes of the blockchain network are based on the concept that no one in the network can be trusted. Proof of work is accepted by nodes to validate any transaction. Proof of work involves doing hefty calculations to find a 32-bit hash value called nonce to solve the mathematical puzzle. The miners create new blocks by abiding by the fact that the transaction volume must be less than 21 million. 21 million is the total number of bitcoins that can be generated. The verified transaction gets a unique identification code and is linked with the previous verified transaction.

Let's understand this with the help of an example-

- Suppose Alice wants to transfer 10 BTC to Bob.
- Now the transaction data of A is shared with the miners from the memory pool. A memory pool is a place where an unconfirmed or unverified transaction waits for its confirmation.
- Miners start competing with themselves to solve the mathematical riddle in order to validate and verify the transaction using proof of work.
- The miner who solves the problem first shares his result with other nodes(miners).
- Once maximum nodes agree with the solution, the transaction block is verified and is then added to the blockchain.
- At the same time, the miner who solved the puzzle gets a reward of 6.25 bitcoins.
- Now, after the addition of the transaction block, the 10 BTC associated with the transaction data is transferred to Bob from Alice



How to check validity of blocks in blockchain

Block validation is a critical process in blockchain technology that ensures the integrity and security of the blockchain by verifying that new blocks of transactions conform to network rules and consensus protocols before being added to the blockchain. This involves checking the validity of transactions, block structure, compliance with the consensus mechanism, and preventing double-spending. Effective block validation is essential for maintaining the accuracy and consistency of the blockchain ledger, securing the network from fraud, and ensuring uniformity across the distributed ledger. As blockchain technology advances, improvements in validation algorithms and consensus mechanisms continue to enhance efficiency and scalability.
Validators are crucial to Blockchain Security as they verify transactions, participate in consensus mechanisms, and produce new blocks. By confirming the validity of transactions and adhering to the network's rules, they prevent fraudulent activities. They also help achieve consensus through mechanisms like Proof of Work or Proof of Stake, making it difficult for malicious actors to alter the blockchain development company. They are incentivized through rewards and penalties, ensuring honest behavior. Their involvement in governance and network upgrades also helps maintain the blockchain's stability and security.

Each block added to the blockchain must satisfy a set of predefined validation rules. If any of these rules are violated, the block is considered invalid and is rejected.

1. Hash Validation
Each block contains a cryptographic hash generated using the block's data. To validate a block:

- The hash of the block is recalculated.
- The recalculated hash is compared with the hash stored in the block.
- If both hashes match, the block data is considered untampered.

This step ensures the integrity of the block's contents.

2. Previous Hash Verification
Every block (except the genesis block) stores the hash of the previous block.

- The previous hash stored in the current block must match the actual hash of the previous block in the chain.
- If this condition fails, the chain is broken and the block is deemed invalid.

This linking mechanism maintains the immutability of the blockchain.

3. Proof of Work Validation
In blockchains using a Proof of Work (PoW) consensus mechanism:

- The block hash must satisfy the network's difficulty requirement (e.g., a specific number of leading zeros).

- This confirms that the miner has performed the required computational work before adding the block.

4. Transaction Validation
All transactions included in a block must be verified before the block is accepted:

- Digital signatures must be valid.
- The sender must have sufficient balance.
- Transactions must follow the correct format.
- Double spending must be prevented.

A block containing invalid transactions is rejected.

5. Timestamp Validation
Each block includes a timestamp indicating when it was created.

- The timestamp must not be in the future.
- It must be greater than the timestamp of the previous block.

This ensures chronological consistency in the blockchain.

6. Block Index Validation
Blocks are ordered sequentially using an index or block height.

- The index of the current block must be exactly one greater than the previous block's index.
- This prevents missing or duplicate blocks in the chain.

7. Genesis Block Verification
The genesis block is the first block in the blockchain and has predefined values.

- Its hash, previous hash, and timestamp must match the known initial configuration.
- Any mismatch indicates an invalid blockchain.

Programs And Outputs:

Endpoints (Can check on Postman) :

- /mine_block → mine a new block
- /get_chain → fetch full blockchain
- /is_valid → verify blockchain
- /add_transaction → add transactions in block

Code :

```
# Importing libraries
import datetime
import hashlib
import json
from flask import Flask, jsonify, request

# ------------------------------------------------------------
# Part 1 - Building the Blockchain
# ------------------------------------------------------------

class Blockchain:

    def __init__(self):
        self.chain = []
        self.transactions = []   # 🔹 Transaction list
        self.create_block(nonce=1, previous_hash='0')

    def create_block(self, nonce, previous_hash):
        block = {
            'index': len(self.chain) + 1,
            'timestamp': str(datetime.datetime.now()),
            'nonce': nonce,
            'transactions': self.transactions,   # 🔹 Add transactions to block
            'previous_hash': previous_hash
        }

        # 🔹 Clear transaction list before mining next block
        self.transactions = []

        self.chain.append(block)
        return block

    def get_previous_block(self):
        return self.chain[-1]

    # ------------------------------------------------------------
    # 🔹 Create Transactions
    # ------------------------------------------------------------
    def create_transaction(self, sender, receiver, amount):
        self.transactions.append({
            'sender': sender,
            'receiver': receiver,
            'amount': amount
        })
```

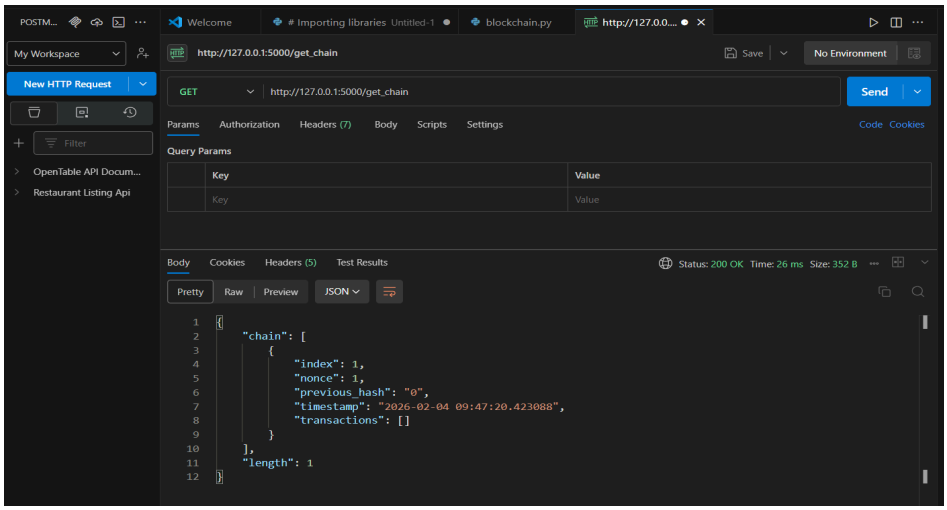
```
    return self.get_previous_block()['index'] + 1

# ------------------------------------------------------------
#  🔹 Proof of Work (Golden Nonce)
# ------------------------------------------------------------
def proof_of_work(self, previous_hash):
    nonce = 0
    while True:
        block_data = str(self.transactions) + previous_hash + str(nonce)
        hash_operation = hashlib.sha256(block_data.encode()).hexdigest()

        #  🔹 Cryptographic puzzle: hash starts with "000"
        if hash_operation[:3] == "000":
            return nonce
        nonce += 1

def hash(self, block):
    encoded_block = json.dumps(block, sort_keys=True).encode()
    return hashlib.sha256(encoded_block).hexdigest()

def is_chain_valid(self, chain):
    previous_block = chain[0]
    index = 1

    while index < len(chain):
        block = chain[index]

        if block['previous_hash'] != self.hash(previous_block):
            return False

        block_data = (
            str(block['transactions']) +
            block['previous_hash'] +
            str(block['nonce'])
        )
        hash_operation = hashlib.sha256(block_data.encode()).hexdigest()

        if hash_operation[:3] != "000":
            return False

        previous_block = block
        index += 1

    return True
```

```
# ------------------------------------------------------------
# Part 2 - Flask API
# ------------------------------------------------------------
app = Flask(__name__)
blockchain = Blockchain()

# ------------------------------------------------------------
# 🔹 Add Transaction API
# ------------------------------------------------------------
@app.route('/add_transaction', methods=['POST'])
def add_transaction():
    json_data = request.get_json()
    required_fields = ['sender', 'receiver', 'amount']

    if not all(field in json_data for field in required_fields):
        return 'Missing fields', 400

    index = blockchain.create_transaction(
        json_data['sender'],
        json_data['receiver'],
        json_data['amount']
    )

    response = {'message': f'Transaction will be added to Block {index}'}
    return jsonify(response), 201

# ------------------------------------------------------------
# 🔹 Mine Block ONLY if transactions exist
# ------------------------------------------------------------
@app.route('/mine_block', methods=['GET'])
def mine_block():

    if len(blockchain.transactions) == 0:
        return jsonify({'message': 'No transactions to mine'}), 400

    previous_block = blockchain.get_previous_block()
    previous_hash = blockchain.hash(previous_block)

    nonce = blockchain.proof_of_work(previous_hash)
    block = blockchain.create_block(nonce, previous_hash)

    response = {
        'message': 'Block mined successfully!',
        'index': block['index'],
        'nonce': block['nonce'],
        'transactions': block['transactions'],
```

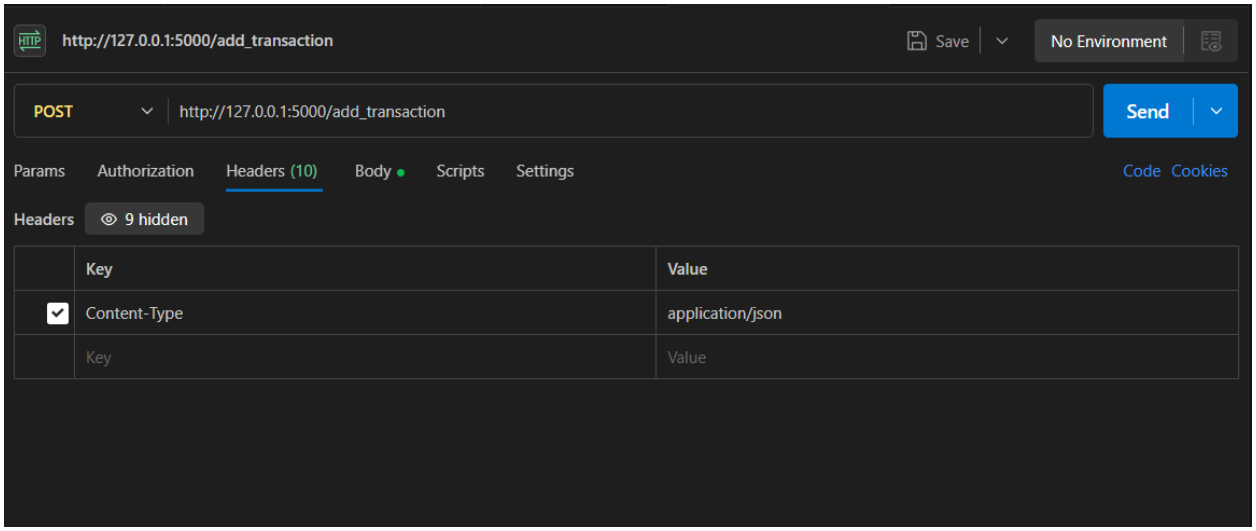
```
        'previous_hash': block['previous_hash']
    }
    return jsonify(response), 200

# ----------------------------------------------------------
# Get full blockchain
# ----------------------------------------------------------
@app.route('/get_chain', methods=['GET'])
def get_chain():
    return jsonify({
        'chain': blockchain.chain,
        'length': len(blockchain.chain)
    }), 200

# ----------------------------------------------------------
# Check blockchain validity
# ----------------------------------------------------------
@app.route('/is_valid', methods=['GET'])
def is_valid():
    if blockchain.is_chain_valid(blockchain.chain):
        return jsonify({'message': 'Blockchain is valid'}), 200
    else:
        return jsonify({'message': 'Blockchain is invalid'}), 400

# ----------------------------------------------------------
# Run Flask App
# ----------------------------------------------------------
app.run(host='0.0.0.0', port=5000, debug=True)
```

Output :

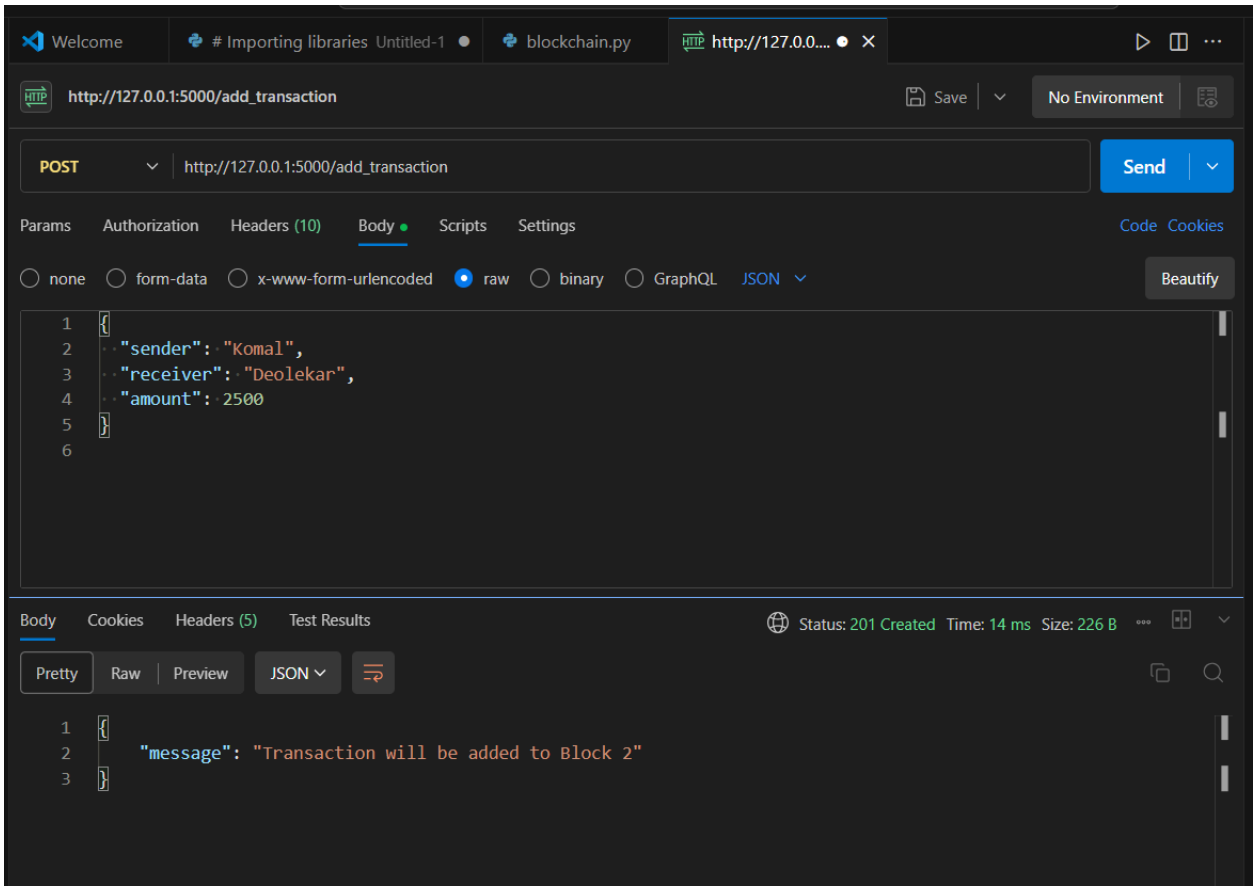
Get_chain : Displaying Initial Chain

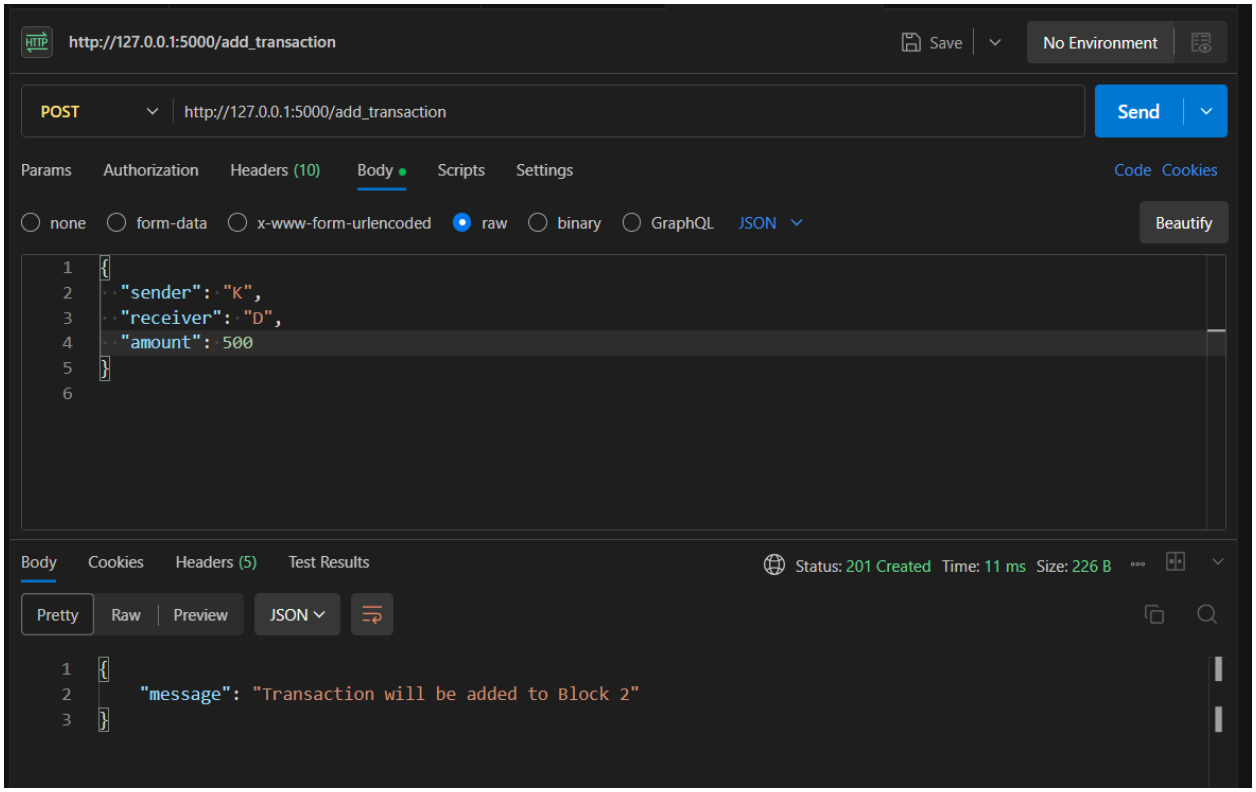


Add_transaction : For adding transactions in transaction pool.
For sending json data add Content-Type : application/json in header of the POST request and in body tab select json and then add key values that you want to send

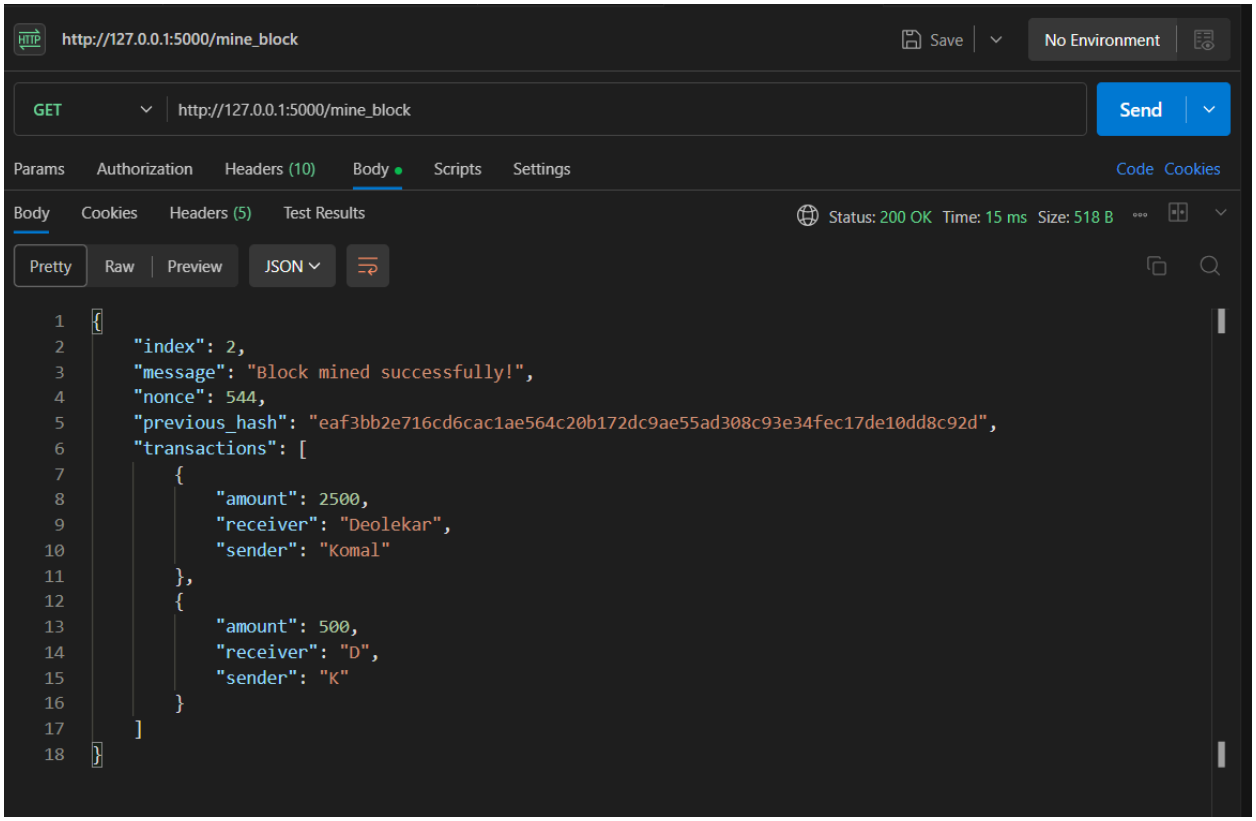


Below transactions will get mined in block 2 as next block which is going to get mined is block 2

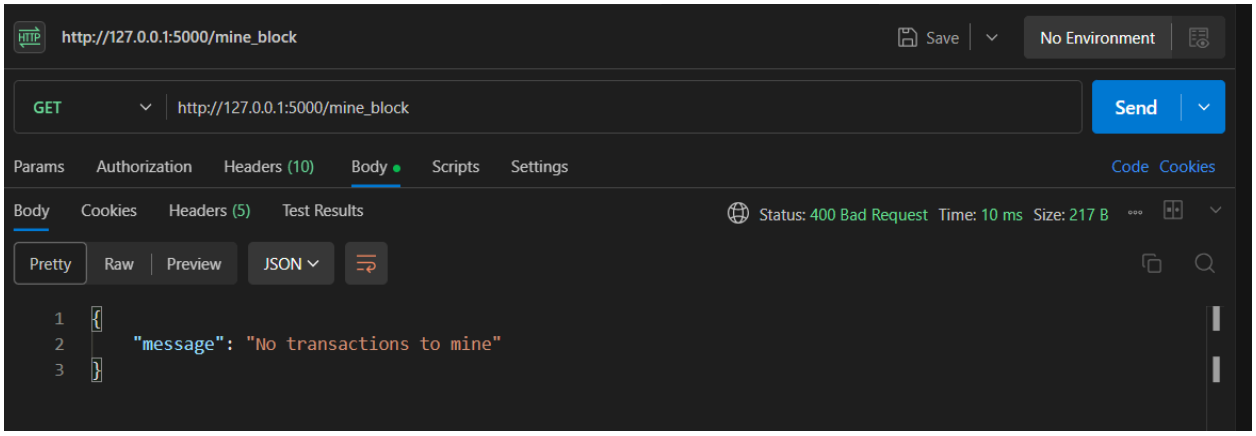




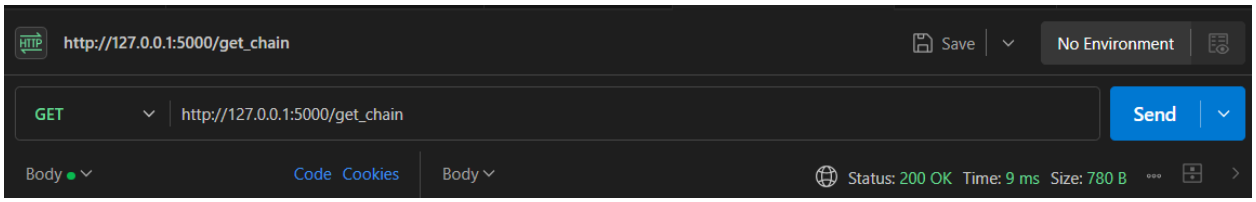
Mine_block : To add block in blockchain



If no transactions are there then block will not get mined

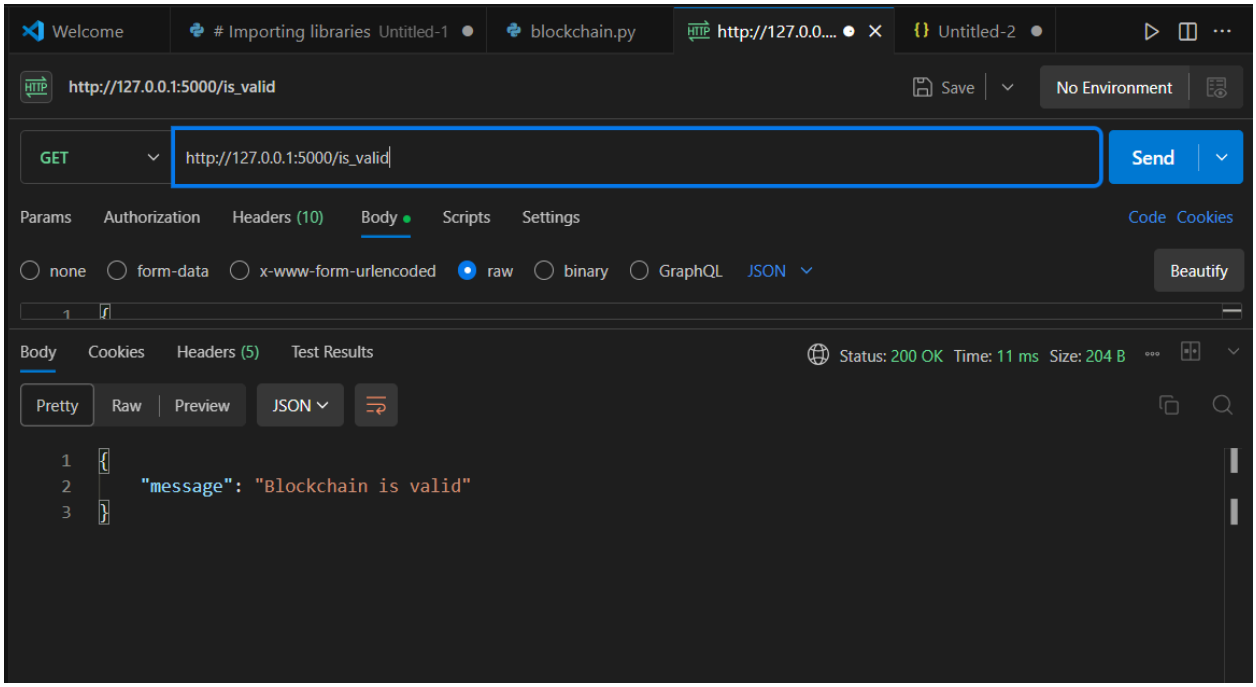


Chain after mining blocks




```
{
  "chain": [
    {
      "index": 1,
      "nonce": 1,
      "previous_hash": "0",
      "timestamp": "2026-02-04 09:47:20.423088",
      "transactions": []
    },
    {
      "index": 2,
      "nonce": 544,
      "previous_hash": "eaf3bb2e716cd6cac1ae564c20b172dc9ae55ad308c93e34fec17de10dd8c92d",
      "timestamp": "2026-02-04 09:59:39.296361",
      "transactions": [
        {
          "amount": 2500,
          "receiver": "Deolekar",
          "sender": "Komal"
        },
        {
          "amount": 500,
          "receiver": "D",
          "sender": "K"
        }
      ]
    }
  ],
  "length": 2
}
```

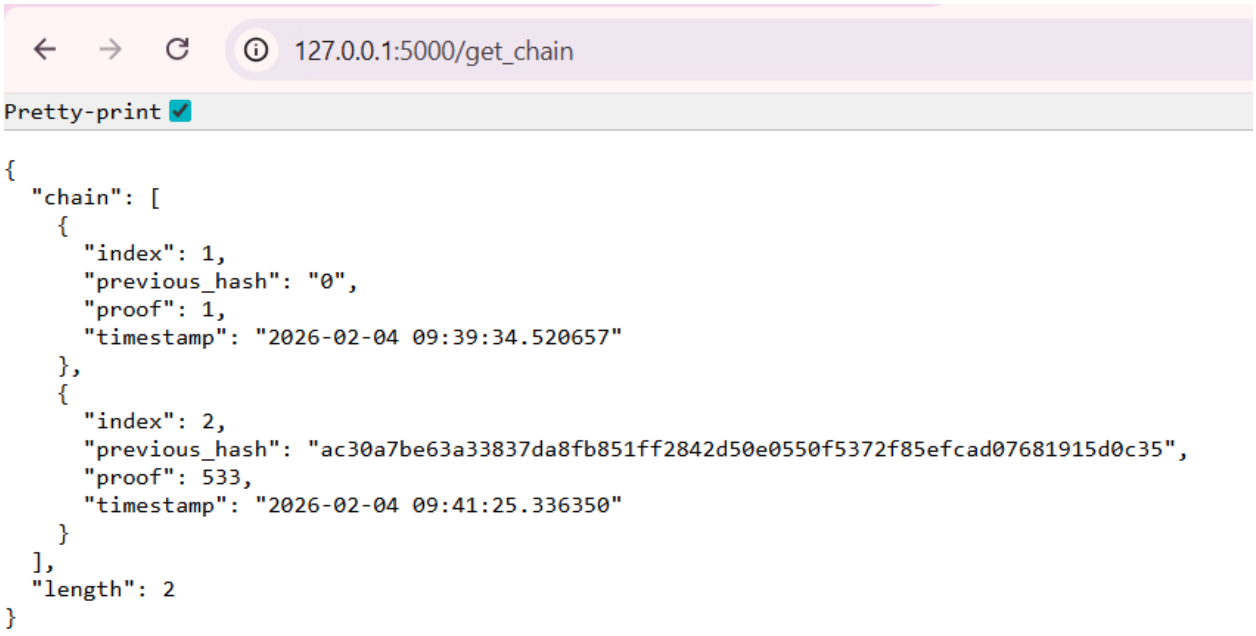
Is_valid : check if chain is valid or not

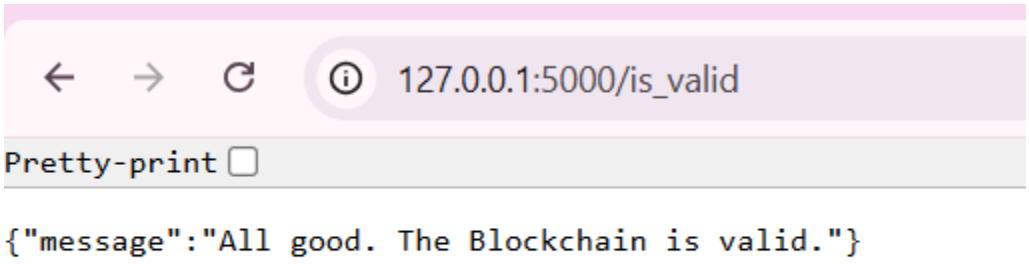


Output on browser

{"chain":[{"index":1,"previous_hash":"0","proof":1,"timestamp":"2026-02-04 09:39:34.520657"}],"length":1}

{
"index": 2,
"message": "Congratulations, you just mined a block!",
"previous_hash": "ac30a7be63a33837da8fb851ff2842d50e0550f5372f85efcad07681915d0c35",
"proof": 533,
"timestamp": "2026-02-04 09:41:25.336350"
}





Conclusion :

In this experiment, a basic blockchain was successfully created using Python. The implementation demonstrated block creation, hashing, and validation using previous hash linkage and Proof of Work. This helped in understanding the core concepts of blockchain such as immutability, data integrity, and secure transaction storage.